UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/693,854	10/24/2003	Rob Relyea	MS1-1780US	3939

22801 7590 11/17/2009

LEE & HAYES, PLLC
601 W. RIVERSIDE AVENUE
SUITE 1400
SPOKANE, WA 99201

EXAMINER	
WANG, BEN C	

ART UNIT	PAPER NUMBER
2192	

NOTIFICATION DATE	DELIVERY MODE
11/17/2009	ELECTRONIC

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Notice of the Office communication was sent electronically on above-indicated "Notification Date" to the following e-mail address(es):

lhptoms@leehayes.com

Office Action Summary	Application No.	Applicant(s)	
	10/693,854	RELYEA ET AL.	
	Examiner	Art Unit	
	BEN C. WANG	2192	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE *3* MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *30 September 2009*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1,4-8,10-14,16,17 and 19-25* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1, 4-8, 10-14, 16, 17, and 19-25* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☒ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date *10/30/2009*.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .
5) ☐ Notice of Informal Patent Application
6) ☐ Other: _____.

DETAILED ACTION

1. A request for continued examination under 37 CFR 1.114, including the fee set forth in 37 CFR 1.17(e), was filed in this application after final rejection. Since this application is eligible for continued examination under 37 CFR 1.114, and the fee set forth in 37 CFR 1.17(e) has been timely paid, the finality of the previous Office action has been withdrawn pursuant to 37 CFR 1.114. Applicant's submission filed on September 30, 2009 has been entered.

2. Applicant's amendment dated September 30, 2009, responding to the Office action mailed July 30, 2009 provided in the rejection of claims 1, 4-8, 10-14, 16, 17, and 19-25, wherein claims 1, 17, and 23 have been amended.

Claims 1, 4-18, 10-14, 16, 17, and 19-25 remain pending in the application and which have been fully considered by the examiner.

Applicant's arguments with respect to claims currently amended have been fully considered but are moot in view of the new grounds of rejection – see *Rubin et al.* - art made of record, as applied hereto.

Information Disclosure Statement

3. The information disclosure statement (IDS) submitted on October 30, 2009 was filed after the mailing date of the Office action on July 30, 2009. The submission is in compliance with the provisions of 37 CFR 1.97. Accordingly, the information disclosure statement is being considered by the examiner.

Claim Rejections – 35 USC § 103(a)

The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

4. Claims 1, 4-8, 10-14, 16, 17, and 19-25 are rejected under 35 U.S.C. 103(a) as being unpatentable over Smith et al. (Pub. No. US 2003/0028685 A1) (hereinafter 'Smith') in view of Stoakley et al. (Pub. No. US 2001/0045961 A1) (hereinafter 'Stoakley'), Priya Lakshminarayanan (*The .NET Schema Object Model, December 04, 2002, XML.com - O'Reily Media, Inc.*) (hereinafter 'Lakshminarayanan'), Powers et al. (*Visual Basic® Programmer's Guide to the .NET Framework Class Library*) (hereinafter 'Powers') and further in view of Rubin et al. (Pat. No. 5,862,379) (hereinafter 'Rubin' - art made of record)

5. **As to claim 1** (Currently Amended), Smith discloses a programming interface embodied on one or more computer readable media having computer-executable instructions for performing steps, comprising:

- generating graphical objects using a first group of services (i.e., P.11, *System.Drawing*; P. 12 – *System.Web.UI*); and
- binding element to data sources, data source classes, and data specific implementation of data collections using a fourth group of services (e.g., Fig. 3, element 204 – Data and XML; [0057], Lines 9-13 - … data binding

functionality …; [0063] – a data namespace ("System.Data") contains classes that enable developers to build components that efficiently manager data from multiple data sources … a collection of types used to access a data source …)

Further, Smith discloses that an application program interface (API) provides a set of functions for application developers who build Web applications on Microsoft Corporation's .NET™ platform (e.g., Abstract) but does not explicitly disclose other limitations stated below.

However, in an analogous art of *System and Theme File Format for Creating Visual Styles*, Stoakley discloses formatting content using a second group of services, wherein the second group of services arranges the graphical objects (e.g., [0013] - …. Specifies the set of drawing, layout and measuring properties and values that are to be applied to each control or graphical component ...; [0039] - … the layout of the parts of a control can be defined as desired. For example, a scroll bar could be rendered with the up and down arrow buttons adjacent to each other at the bottom of the scroll bar, instead of the up arrow and the top and the down arrow at the bottom … Each different appearance setting is coordinated and is called a "them." …; [0047] – Drawing APIs 218 are generally used to render and assist in the layout of the needed control parts and other components …)

Therefore, it would have been obvious to one of ordinary skill in the art, at the time the invention was made to combine the teachings of Stoakley into the

Smith's system to further provide other limitations stated above in the Smith system.

The motivation is that it would further enhance the Smith's system by taking, advancing and/or incorporating the Stoakley's system which offers significant advantages of an effective system, method and file format that allows a designer to efficiently create a visual style for a set of controls and graphical components as once suggested by Stoakley (e.g., [0009])

Furthermore, Stoakley discloses a method of specifying a visual style for a set of graphical components is provided for use on a computer system having a graphical operating environment (e.g., Abstract) but Smith and Stoakley do not explicitly disclose other limitations stated below.

However, in an analogous art of *The .NET® Schema Object Model*, Lakshminarayanan discloses the followings:

- using a common markup language to map classes and properties specified in the markup language to an instantiated tree of objects across the first group of services, the second group of services, the third group of services, and the fourth group of services; and
- integrating the first group of services, the second group of and the third group of services, and the fourth group of services using a consistent programming model and consistent services across the four service groups (e.g., P. 1, 1st Para – This article focuses on an API in the .NET® platform, the XML Schema Object Model (SOM), SOM is rich API which allows developers to create, edit, and validate schemas programmatically

– on of the few such tools available so far; .2nd Para – SOM operates on schema documents analogously to the way DOM operates on XML documents. Schema documents are valid XML files that, once loaded into the SOM, convey meaning about the structure and validity of other XML documents which conform to the schema. SOM is indispensable for a certain class of application, like a schema editor, where it needs to construct the schema in memory and check the schema's validity according to the WXS (W3C XML Schema) specifications; 4th Para – This mapping helps easy use of the API. For a complete listing of all the classes available in the System.Xml.Schema namespace, refer to the .NET® Framework Class Library Reference)

Therefore, it would have been obvious to one of ordinary skill in the art, at the time the invention was made to combine the teachings of Lakshminarayanan into the Smith-Stoakley's system to further provide other limitations stated above in the Smith-Stoakley system.

The motivation is that it would further enhance the Smith-Stoakley's system by taking, advancing and/or incorporating Lakshminarayanan's system which offers significant advantages that on an API in the .NET® platform, the XML Schema Object Model (SOM), SOM is a rich API which allows developers to create, edit, and validate schema programmatically – one of the few such tools available so far as once suggested by Lakshminarayanan (e.g., P. 1, 1st Para)

Furthermore, Smith discloses that an application program interface (API) provides a set of functions for application developers who build Web applications

on Microsoft Corporation's .NET™ platform, but Smith, Stoakley and Lakshminarayanan do not explicitly discloses wherein the fourth group of services further handle exceptions in data entry.

However, in an analogous art of *Visual Basic® Programmer's Guide to the .NET Framework Class Library*, Powers discloses wherein the fourth group of services further handle exceptions in data entry (e.g., Chapter 15. Data Storage and Access, Table 15.1 – Key Classes Covered, 4th entry – *ConstraintException* and 7th entry – *DataException*)

Therefore, it would have been obvious to one of ordinary skill in the art, at the time the invention was made to combine the teachings of Powers into the Smith-Stoakley-Lakshminarayanan's system to further provide the fourth group of services further handle exceptions in data entry in the Smith-Stoakley-Lakshminarayanan system.

The motivation is that it would further enhance the Smith-Stoakley-Lakshminarayanan's system by taking, advancing and/or incorporating the Powers' system which offers significant advantages for using data access exceptions within .NET Framework Class Library as once suggested by Powers (e.g., Chap. 15 – Data Storage and Access Key Classes Related to Data)

Moreover, Smith, Stoakley, Lakshminarayanan, and Powers do not explicitly disclose the limitations stated below.

However, in an analogous art of *Visual Programming Tool for Developing Software Applications*, Rubin discloses creating components of the graphical objects using a third group of services, wherein the first group of services

includes a service that customizes a behavior of the graphical objects to style and/or theme the graphical objects, the service including interfaces for controlling a behavior of a menu item (e.g., Fig. 7 – elements 100, 104; Col. 9, Lines 26-64 – As shown in FIG. 7, using the tools of the Toolbox, a user has drawn a command 100 in a application window 102 … Selecting one of the menu items …; App – 15, Properties: …. Use event procedures to set list box properties at run time …; App – 16, Predefined link behaviors: The user can use predefined link behaviors to respond to list box events ...; App – 17 - Event – The user can create scripts and links to respond to the following list box events - Select), for controlling a visibility behavior of a scroll bar (e.g., APPENDIX 8 – Scroll Bar, Properties: **Visible;** App - 22, As the destination – **ToggleVisibility**), and for controlling selection behavior of a listbox (e.g., APPENDIX 6 – List Box - … The user can use list boxes to display a list of items from which the user can choose one or more items)

Therefore, it would have been obvious to one of ordinary skill in the art, at the time the invention was made to combine the teachings of Rubin into the Smith-Stoakley-Lakshminarayanan-Powers' system to further provide the limitations stated above in the Smith-Stoakley-Lakshminarayanan-Powers system.

The motivation is that it would further enhance the Smith-Stoakley-Lakshminarayanan's system by taking, advancing and/or incorporating the Rubin's system which offers significant advantages for overcoming standing significant barriers to users who are not experienced in programming or writing code as once suggested by Rubin (e.g., Col. 1, Lines 31-37)

6. **As to claim 4**, (Previously Presented) (incorporating the rejection in claim 1), Smith discloses a programming interface wherein the first group of services, the second group of services, the third group of services, and fourth group of services share a common event system (e.g., [0045] – event handling; [0049], Lines 7-10; [0069])

7. **As to claim 5**, (Previously Presented) (incorporating the rejection in claim 1), Smith discloses a programming interface wherein the first group of services, the second group of services, the third group of services, and the fourth group of services share a common property definition system (e.g., [0049], Lines 7-10; [0075]; [0079], Lines 1-10)

8. **As to claim 6**, (Previously Presented) (incorporating the rejection in claim 1), Smith discloses a programming interface wherein the first group of services, the second group of services, the third group of services, and the fourth group of services share a common input paradigm (e.g., [0092], Lines 6-10; [0088], Lines 4-7; [0093], Lines 3-7).

9. **As to claim 7**, (Previously Presented) (incorporating the rejection in claim 1), Smith discloses a programming interface wherein the first group of services, the second group of services, the third group of services, and the fourth group of services share a common system for nesting elements associated with a

particular group of services within elements associated with another group of services (e.g., Fig. 3; [0052] through [0059])

10. **As to claim 8**, (Original) (incorporating the rejection in claim 1), Smith discloses a programming interface wherein the first group of services includes a service that determines an appearance of the graphical objects (e.g., [0030], Lines 4-8 – HTML defines how elements are displayed)

11. **As to claim 10**, (Original) (incorporating the rejection in claim 1), Smith discloses a programming interface wherein the first group of services includes a service that determines an arrangement of the graphical objects (e.g., [0030], Lines 4-8 – HTML defines how elements are displayed)

12. **As to claim 11**, (Original) (incorporating the rejection in claim 1), Smith discloses a programming interface wherein the first group of services includes a plurality of nested elements that define the graphical objects (e.g., Fig. 3, element 312 – UI; [0057])

13. **As to claim 12**, (Original) (incorporating the rejection in claim 1), Smith discloses a programming interface wherein the objects are comprised of one or more elements defined by vector graphical graphics (e.g., [0062] – vector graphics functionality)

14. **As to claim 13** (Original) (incorporating the rejection in claim 1), Smith discloses a programming interface and an application program interface wherein the first group of services can define window properties in a markup language without launching a new window (e.g., [0061] – A windows forms namespace ("*System.Windows.Forms*") containing classes for creating Windows®-based client applications)

15. **As to claim 14**, (Original) (incorporating the rejection in claim 1), Smith discloses a programming interface wherein the first group of services generates a user interface containing a plurality of graphical objects (i.e., Fig. 3, element 202 – Client Application; [0048] – the client application namespace pertains to drawing and client side UI functionality; P. 13, Left-Col., Lines 50-52; P. 14, Left-Col., Lines 17-19, Right-Col., Lines 18-20; P. 15, Left-Col., Lines 20-22, Right-Col., Lines 37-39; P. 16, Left-Col. Lines 24-27, Right-Col., Lines 5-8, 34-36)

16. **As to claim 16**, (Original) (incorporating the rejection in claim 1), Smith discloses a software architecture comprising the programming interface (e.g., Fig. 2; [0022]; [0044], Lines 1-30)

17. **As to claim 17** (Currently Amended), Smith discloses an application program interface embodied on one or more computer readable media having computer-executable instructions for performing steps, comprising:

- creating components of the graphical objects using a third group of services (i.e., Fig. 3, element 202 – Client Application; [0048] – the client application namespace pertains to drawing and client side UI functionality; P. 13, Left-Col., Lines 50-52; P. 14, Left-Col., Lines 17-19, Right-Col., Lines 18-20; P. 15, Left-Col., Lines 20-22, Right-Col., Lines 37-39; P. 16, Left-Col. Lines 24-27, Right-Col., Lines 5-8, 34-36), wherein the first group of services, the second group of services, the third group of services, and the fourth group of services are integrated via: sharing a common programming model (e.g., Fig. 3; [0023]; [0052], Lines 1-6);
- binding elements to data sources, data source classes, and data specific implementations of data collections using a fourth group of services (e.g., Fig. 3, element 204 – Data and XML; [0057], Lines 9-13 - … data binding functionality …; [0063] – a data namespace ("System.Data") contains classes that enable developers to build components that efficiently manager data from multiple data sources … a collection of types used to access a data source …)

Further, Smith discloses that an application program interface (API) provides a set of functions for application developers who build Web applications on Microsoft Corporation's .NET™ platform (e.g., Abstract) but does not explicitly disclose other limitations stated below.

However, in an analogous art of *System and Theme File Format for Creating Visual Styles*, Stoakley discloses formatting content using a second group of services, wherein the second group of services arranges the graphical objects

(e.g., [0013] - …. Specifies the set of drawing, layout and measuring properties and values that are to be applied to each control or graphical component ...; [0039] - … the layout of the parts of a control can be defined as desired. For example, a scroll bar could be rendered with the up and down arrow buttons adjacent to each other at the bottom of the scroll bar, instead of the up arrow and the top and the down arrow at the bottom … Each different appearance setting is coordinated and is called a "them." …; [0047] – Drawing APIs 218 are generally used to render and assist in the layout of the needed control parts and other components …)

Therefore, it would have been obvious to one of ordinary skill in the art, at the time the invention was made to combine the teachings of Stoakley into the Smith's system to further provide other limitations stated above in the Smith system.

The motivation is that it would further enhance the Smith's system by taking, advancing and/or incorporating the Stoakley's system which offers significant advantages of an effective system, method and file format that allows a designer to efficiently create a visual style for a set of controls and graphical components as once suggested by Stoakley (e.g., [0009])

Furthermore, Stoakley discloses a method of specifying a visual style for a set of graphical components is provided for use on a computer system having a graphical operating environment (e.g., Abstract) but Smith and Stoakley do not explicitly disclose other limitations stated below.

However, in an analogous art of *The .NET® Schema Object Model*, Lakshminarayanan discloses using a common markup language across the three services to map classes and properties specified in the markup language to an instantiated tree of objects (e.g., P. 1, 1st Para – This article focuses on an API in the .NET® platform, the XML Schema Object Model (SOM), SOM is rich API which allows developers to create, edit, and validate schemas programmatically – on of the few such tools available so far; .2nd Para – SOM operates on schema documents analogously to the way DOM operates on XML documents. Schema documents are valid XML files that, once loaded into the SOM, convey meaning about the structure and validity of other XML documents which conform to the schema. SOM is indispensable for a certain class of application, like a schema editor, where it needs to construct the schema in memory and check the schema's validity according to the WXS (W3C XML Schema) specifications; 4th Para – This mapping helps easy use of the API. For a complete listing of all the classes available in the System.Xml.Schema namespace, refer to the .NET® Framework Class Library Reference)

Therefore, it would have been obvious to one of ordinary skill in the art, at the time the invention was made to combine the teachings of Lakshminarayanan into the Smith-Stoakley's system to further provide using a common markup language across the three services to map classes and properties specified in the markup language to an instantiated tree of objects in the Smith-Stoakley system.

The motivation is that it would further enhance the Smith-Stoakley's system by taking, advancing and/or incorporating Lakshminarayanan's system which offers significant advantages that on an API in the .NET® platform, the XML Schema Object Model (SOM), SOM is a rich API which allows developers to create, edit, and validate schema programmatically – one of the few such tools available so far as once suggested by Lakshminarayanan (e.g., P. 1, 1st Para)

Furthermore, Smith discloses that an application program interface (API) provides a set of functions for application developers who build Web applications on Microsoft Corporation's .NET™ platform (e.g., Abstract), but Smith, Stoakley and Lakshminarayanan do not explicitly discloses wherein the fourth group of services further handle exceptions in data entry.

However, in an analogous art of *Visual Basic® Programmer's Guide to the .NET Framework Class Library*, Powers discloses wherein the fourth group of services further handle exceptions in data entry (e.g., Chapter 15. Data Storage and Access, Table 15.1 – Key Classes Covered, 4th entry – *ConstraintException* and 7th entry – *DataException*)

Therefore, it would have been obvious to one of ordinary skill in the art, at the time the invention was made to combine the teachings of Powers into the Smith-Stoakley-Lakshminarayanan's system to further provide the fourth group of services further handle exceptions in data entry in the Smith-Stoakley-Lakshminarayanan system.

The motivation is that it would further enhance the Smith-Stoakley-Lakshminarayanan's system by taking, advancing and/or incorporating the

Powers' system which offers significant advantages for using data access exceptions within .NET Framework Class Library as once suggested by Powers (e.g., Chap. 15 – Data Storage and Access Key Classes Related to Data)

Moreover, Smith, Stoakley, Lakshminarayanan, and Powers do not explicitly disclose the limitations stated below.

However, in an analogous art of *Visual Programming Tool for Developing Software Applications*, Rubin discloses generating graphical objects using a first group of services, wherein the first group of services includes a service that customizes a behavior of the graphical objects to style and/or theme the graphical objects, the service including interfaces for controlling a behavior of a menu item (e.g., Fig. 7 – elements 100, 104; Col. 9, Lines 26-64 – As shown in FIG. 7, using the tools of the Toolbox, a user has drawn a command 100 in a application window 102 … Selecting one of the menu items …; App – 15, Properties: …. Use event procedures to set list box properties at run time …; App – 16, Predefined link behaviors: The user can use predefined link behaviors to respond to list box events ...; App – 17 - Event – The user can create scripts and links to respond to the following list box events - Select), for controlling a visibility behavior of a scroll bar (e.g., APPENDIX 8 – Scroll Bar, Properties: **Visible;** App - 22, As the destination – **ToggleVisibility**), and for controlling selection behavior of a listbox (e.g., APPENDIX 6 – List Box - … The user can use list boxes to display a list of items from which the user can choose one or more items).

Therefore, it would have been obvious to one of ordinary skill in the art, at the time the invention was made to combine the teachings of Rubin into the Smith-

Stoakley-Lakshminarayanan-Powers' system to further provide the limitations stated above in the Smith-Stoakley-Lakshminarayanan-Powers system.

The motivation is that it would further enhance the Smith-Stoakley-Lakshminarayanan's system by taking, advancing and/or incorporating the Rubin's system which offers significant advantages for overcoming standing significant barriers to users who are not experienced in programming or writing code as once suggested by Rubin (e.g., Col. 1, Lines 31-37)

18. **As to claim 19**, (Original) (incorporating the rejection in claim 17), Smith discloses an application program interface wherein the third group of services includes services to generate geometric shapes (e.g., [0048] – the client applications namespace pertains to drawing and client side UI functionality. It supplies types that enable drawing of two-dimensional, imaging, and printing, as well as the ability to construct window forms, menus, boxes, and so on)

19. **As to claim 20**, (Original) (incorporating the rejection in claim 17), Smith discloses an application program interface wherein the second group of services includes arranging a plurality of data elements (e.g., [0030], Lines 4-8 – XML is used for defining data element on a Web page)

20. **As to claim 21**, (Previously Presented) (incorporating the rejection in claim 17), Smith discloses an application program interface wherein the first group of services includes: a service that determines an appearance of a

graphical object (e.g., [0030], Lines 4-8 – HTML defines how elements are displayed)

21. **As to claim 22**, (Original) (incorporating the rejection in claim 17), please refer to claim **13** as set forth above accordingly.

22. **As to claim 23** (Currently Amended), Smith discloses a computer system including one or more microprocessors (Fig. 4, element 404 – Processing Unit; [0085], Lines 3-5) and one more software programs (Fig. 4, elements 428 - Application Programs, 430 – Program Modules, 432 – Program Data; [0091], Lines 4-5), the one or more software programs utilizing a interface (Fig. 2, element 142 – Application Program Interface; [0039], Lines 1-4) to request services from an operating system (Fig. 2, element 146(1) – Operating System), the services or programming interface including separate commands to request consisting of the following groups of services:

- a third group of services for creating components of the graphical objects (i.e., Fig. 3, element 202 – Client Application; [0048] – the client application namespace pertains to drawing and client side UI functionality; P. 13, Left-Col., Lines 50-52; P. 14, Left-Col., Lines 17-19, Right-Col., Lines 18-20; P. 15, Left-Col., Lines 20-22, Right-Col., Lines 37-39; P. 16, Left-Col. Lines 24-27, Right-Col., Lines 5-8, 34-36); and
- a fourth group of services that bind elements to data sources, data source classes, and data specific implementations of data collections (e.g., Fig. 3,

element 204 – Data and XML; [0057], Lines 9-13 - … data binding functionality …; [0063] – a data namespace ("System.Data") contains classes that enable developers to build components that efficiently manager data from multiple data sources … a collection of types used to access a data source …); and

- wherein the first group of services, the second group of services, and the third group of services are integrated by sharing a common programming model (e.g., Fig. 3; [0023]; [0052], Lines 1-6)

Further, Smith discloses that an application program interface (API) provides a set of functions for application developers who build Web applications on Microsoft Corporation's .NET™ platform (e.g., Abstract) but does not explicitly disclose other limitations stated below.

However, in an analogous art of *System and Theme File Format for Creating Visual Styles*, Stoakley discloses a second group of services for formatting content, wherein the second group of services arrange the graphical objects (e.g., [0013] - …. Specifies the set of drawing, layout and measuring properties and values that are to be applied to each control or graphical component ...; [0039] - … the layout of the parts of a control can be defined as desired. For example, a scroll bar could be rendered with the up and down arrow buttons adjacent to each other at the bottom of the scroll bar, instead of the up arrow and the top and the down arrow at the bottom … Each different appearance setting is coordinated and is called a "them." …; [0047] – Drawing APIs 218 are generally

used to render and assist in the layout of the needed control parts and other components …)

Therefore, it would have been obvious to one of ordinary skill in the art, at the time the invention was made to combine the teachings of Stoakley into the Smith's system to further provide other limitations stated above in the Smith system.

The motivation is that it would further enhance the Smith's system by taking, advancing and/or incorporating the Stoakley's system which offers significant advantages of an effective system, method and file format that allows a designer to efficiently create a visual style for a set of controls and graphical components as once suggested by Stoakley (e.g., [0009])

Furthermore, Stoakley discloses a method of specifying a visual style for a set of graphical components is provided for use on a computer system having a graphical operating environment but Smith and Stoakley do not explicitly disclose other limitations stated below.

However, in an analogous art of *The .NET® Schema Object Model*, Lakshminarayanan discloses consistent services and using a common markup language to map classes and properties specified in the markup language to an instantiated tree of objects across the first, second, and third group of services (e.g., P. 1, 1st Para – This article focuses on an API in the .NET® platform, the XML Schema Object Model (SOM), SOM is rich API which allows developers to create, edit, and validate schemas programmatically – on of the few such tools available so far; .2nd Para – SOM operates on schema documents analogously to

the way DOM operates on XML documents. Schema documents are valid XML files that, once loaded into the SOM, convey meaning about the structure and validity of other XML documents which conform to the schema. SOM is indispensable for a certain class of application, like a schema editor, where it needs to construct the schema in memory and check the schema's validity according to the WXS (W3C XML Schema) specifications; 4th Para – This mapping helps easy use of the API. For a complete listing of all the classes available in the System.Xml.Schema namespace, refer to the .NET® Framework Class Library Reference)

Therefore, it would have been obvious to one of ordinary skill in the art, at the time the invention was made to combine the teachings of Lakshminarayanan into the Smith-Stoakley's system to further provide consistent services and using a common markup language to map classes and properties specified in the markup language to an instantiated tree of objects across the first, second, and third group of services in the Smith-Stoakley system.

The motivation is that it would further enhance the Smith-Stoakley's system by taking, advancing and/or incorporating Lakshminarayanan's system which offers significant advantages that on an API in the .NET® platform, the XML Schema Object Model (SOM), SOM is a rich API which allows developers to create, edit, and validate schema programmatically – one of the few such tools available so far as once suggested by Lakshminarayanan (e.g., P. 1, 1st Para)

Furthermore, Smith discloses that an application program interface (API) provides a set of functions for application developers who build Web applications

on Microsoft Corporation's .NET™ platform, but Smith, Stoakley and Lakshminarayanan do not explicitly discloses wherein the fourth group of services further handle exceptions in data entry.

However, in an analogous art of *Visual Basic® Programmer's Guide to the .NET Framework Class Library*, Powers discloses wherein the fourth group of services further handle exceptions in data entry (e.g., Chapter 15. Data Storage and Access, Table 15.1 – Key Classes Covered, 4th entry – *ConstraintException* and 7th entry – *DataException*)

Therefore, it would have been obvious to one of ordinary skill in the art, at the time the invention was made to combine the teachings of Powers into the Smith-Stoakley-Lakshminarayanan's system to further provide the fourth group of services further handle exceptions in data entry in the Smith-Stoakley-Lakshminarayanan system.

The motivation is that it would further enhance the Smith-Stoakley-Lakshminarayanan's system by taking, advancing and/or incorporating the Powers' system which offers significant advantages for using data access exceptions within .NET Framework Class Library as once suggested by Powers (e.g., Chap. 15 – Data Storage and Access Key Classes Related to Data)

Moreover, Smith, Stoakley, Lakshminarayanan, and Powers do not explicitly disclose the limitations stated below.

However, in an analogous art of *Visual Programming Tool for Developing Software Applications*, Rubin discloses a first group of services for generating graphical objects, wherein the first group of services includes a service that

customizes a behavior of the graphical objects to style and/or theme the graphical objects, the service including interfaces for controlling a behavior of a menu item (e.g., Fig. 7 – elements 100, 104; Col. 9, Lines 26-64 – As shown in FIG. 7, using the tools of the Toolbox, a user has drawn a command 100 in a application window 102 … Selecting one of the menu items …; App – 15, Properties: …. Use event procedures to set list box properties at run time …; App – 16, Predefined link behaviors: The user can use predefined link behaviors to respond to list box events ...; App – 17 - Event – The user can create scripts and links to respond to the following list box events - Select), for controlling a visibility behavior of a scroll bar (e.g., APPENDIX 8 – Scroll Bar, Properties: **Visible;** App - 22, As the destination – **ToggleVisibility**), and for controlling selection behavior of a listbox (e.g., APPENDIX 6 – List Box - … The user can use list boxes to display a list of items from which the user can choose one or more items).

Therefore, it would have been obvious to one of ordinary skill in the art, at the time the invention was made to combine the teachings of Rubin into the Smith-Stoakley-Lakshminarayanan-Powers' system to further provide the limitations stated above in the Smith-Stoakley-Lakshminarayanan-Powers system.

The motivation is that it would further enhance the Smith-Stoakley-Lakshminarayanan's system by taking, advancing and/or incorporating the Rubin's system which offers significant advantages for overcoming standing significant barriers to users who are not experienced in programming or writing code as once suggested by Rubin (e.g., Col. 1, Lines 31-37)

23. **As to claim 24**, (Previously Presented) (incorporating the rejection in claim 23), Smith discloses a computer system wherein the first group of services includes: a service for defining an appearance of the graphical objects (e.g., [0030], Lines 4-8 – HTML defines how elements are displayed) (e.g., Fig. 3, element 202 – Client Application; [0048] – the client application namespace pertains to drawing and client side UI functionality; P. 13, Left-Col., Lines 50-52; P. 14, Left-Col., Lines 17-19, Right-Col., Lines 18-20; P. 15, Left-Col., Lines 20-22, Right-Col., Lines 37-39; P. 16, Left-Col. Lines 24-27, Right-Col., Lines 5-8, 34-36)

24. **As to claim 25**, (Previously Presented) (incorporating the rejection in claim 23), Smith discloses a computer system wherein the third group of services includes services to generate a plurality of geometric shapes (e.g., [0048] – the client applications namespace pertains to drawing and client side UI functionality. It supplies types that enable drawing of two-dimensional, imaging, and printing, as well as the ability to construct window forms, menus, boxes, and so on)

Conclusion

25. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Ben C. Wang whose telephone number is 571-270-1240. The examiner can normally be reached on Monday - Friday, 8:00 a.m. - 5:00 p.m., EST.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Tuan Q. Dam can be reached on 571-272-3695. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

/Ben C Wang/
Ben C. Wang
Examiner, Art Unit 2192